Question **1**

Correct

Note de 1,50 sur 1,50

**Cette question devrait être faite sans utiliser un compilateur.**

On désire déterminer l'ordre de construction des différents objets/sous-objets pour une déclaration donnée. On ne listera pas dans cet ordre la "construction" des types fondamentaux, qui n'ont pas de constructeur.

Soient les définitions de classes suivantes (dans des fichiers séparés et on suppose que les "include" sont faits correctement pour la compilation):

```
class A {
public:
   A() { }
};
```

```
class B : public D, public A {
public:
   B() { }
private:
   A att1_;
   D* att2_;
};
```

```
class C : public A {
public:
   C() { att2_ = new B(); }
private:
   D att1_;
   B* att2_;
};
```

```
class D {
public:
   D() { }
};
```

Soit le programme suivant:

```
int main() {
   C x;
}
```

On veut connaître l'ordre de construction lorsque l'on exécute ce programme. Indice: il y a 7 objets/sous-objets construits.

Indiquez uniquement les noms des classes dans le bon ordre, exemple: A B C D A B C

**Réponse :** (régime de pénalités : 0 %)

Indiquez l'ordre de début d'exécution du corps des constructeurs pour les classes ci-haut:

A D C D A A B

Réponse bien enregistrée: ADCDAAB

*Ce message vert n'indique pas si la réponse est bonne ou non, seulement que toutes les cases ont été remplies:*

Tous les tests ont été réussis ! ✔

Correct

Note pour cet envoi : 1,50/1,50.

Question **2**

Correct

Note de 3,00 sur 3,00

Écrivez la définition complète de la classe `Etudiant` qui servira comme classe de base pour plusieurs formes d'étudiant.

Cette classe contient les élements suivants :
- Un attribut `matricule_` de type `int`.
- Un attribut `programme_` de type `string`. Cet attribut doit être seulement accessible aux classes dérivées et à la classe elle-même.
- Un constructeur qui prend en paramètres le matricule et le programme (dans cet ordre) et les initialise.
- Les méthodes non virtuelles `getMatricule` et `getProgramme` qui retournent les valeurs de `matricule_` et `programme_`.
- Une méthode virtuelle pure `changerProgramme` qui prend en paramètre un nouveau programme (une string par référence constante) et ne retourne rien.

Donnez la définition de cette classe. Si des méthodes sont à définir, vous pouvez le faire à l'intérieur ou à l'extérieur de la classe, comme vous voulez.

**Par exemple:**

| Test | Résultat |
|---|---|
| `// Implémentation correcte des méthodes.`<br>`EtudiantTest test(1337, "googoo");`<br>`const Etudiant& etud = test;`<br>`cout << etud.getMatricule() << ", " << etud.getProgramme() << endl;`<br>`test.changerProgramme("gaga");`<br>`cout << etud.getMatricule() << ", " << etud.getProgramme() << endl;` | `1337, googoo`<br>`1337, gaga` |

**Réponse :** (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1  class Etudiant {
2  public:
3      int getMatricule() const {
4          return matricule_;
5      }
6      virtual void changerProgramme(const string& nouveauProgr
7      Etudiant(int matricule, const string& programme) : matri
8      virtual~ Etudiant()=default;
9      string getProgramme() const {
10         return programme_;
11     }
12 protected:
13     string programme_;
```

```
14  private:
15      int matricule_;
16  };
```

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✔ | `// La classe est abstraite`<br>`cout << is_abstract_v<Etudiant> << endl;` | `true` | `true` | ✔ |
| ✔ | `// La classe a la bonne taille (la bonne quantité de données)`<br>`cout << (sizeof(Etudiant) == tailleEtudiantAttendue) << endl;` | `true` | `true` | ✔ |
| ✔ | `// matricule_ et programme_ ne sont pas accessibles publiquement`<br>`cout << has_public_member_matricule_<Etudiant>() << endl;`<br>`cout << has_public_member_programme_<Etudiant>() << endl;` | `false`<br>`false` | `false`<br>`false` | ✔ |
| ✔ | `// On peut en faire un unique_ptr et`<br>`// la destruction se fait correctement.`<br>`unique_ptr<Etudiant> ptr;`<br>`cout << permetUniquePtr << endl;` | `true` | `true` | ✔ |
| ✔ | `// Réussi à instancier une classe dérivée et méthode virtuelle correcte`<br>`EtudiantTest test(69420, "heyyyy");`<br>`cout << has_virtual_changerProgramme_v<EtudiantTest> << endl;` | `true` | `true` | ✔ |
| ✔ | `// Implémentation correcte des méthodes.`<br>`EtudiantTest test(1337, "googoo");`<br>`const Etudiant& etud = test;`<br>`cout << etud.getMatricule() << ", " << etud.getProgramme() << endl;`<br>`test.changerProgramme("gaga");`<br>`cout << etud.getMatricule() << ", " << etud.getProgramme() << endl;` | `1337, googoo`<br>`1337, gaga` | `1337, googoo`<br>`1337, gaga` | ✔ |

Tous les tests ont été réussis ! ✔

## Solution de l'auteur de la question (Cpp):

```cpp
class Etudiant {
public:
    Etudiant(int matricule, const string& programme)
    : matricule_(matricule),
      programme_(programme) { }

    virtual ~Etudiant() = default;

    int getMatricule() const {
        return matricule_;
    }
    const string& getProgramme() const {
        return programme_;
    }
    virtual void changerProgramme(const string& programme

protected:
    string programme_;

private:
```

Correct

Note pour cet envoi : 3,00/3,00.

Question **3**

Incorrect

Note de 0,00 sur 2,50

Écrivez la définition complète de la classe `EtudiantCyclesSup` qui hérite de `Etudiant` et qui représente un étudiant gradué. Un étudiant gradué a un superviseur représenté par la classe `Professeur`. Cette classe est déjà définie pour vous.

Cette classe contiendra les élements suivants:
- Un attribut `superviseur_` de type pointeur de `Professeur`.
- Un constructeur qui prend en paramètres le matricule, le programme et le superviseur (dans cet ordre) et qui les initialise. N'oubliez pas que `Etudiant` possède déjà un constructeur qui initialise le matricule et le programme.
- Les méthodes non virtuelles `getSuperviseur` et `setSuperviseur` qui retourne et change le superviseur, respectivement.
- Une redéfinition de la méthode `changerProgramme` qui affiche `"pls halp"` (avec retour de ligne) et change la valeur de `programme_` avec ce qui est passé en paramètre.

Donnez la définition de cette classe.  Si des méthodes sont à définir, vous pouvez le faire à l'intérieur ou à l'extérieur de la classe, comme vous voulez.

NOTE: Cette question n'utilise pas votre version de la classe `Etudiant` de la question précédente et peut donc être répondue sans avoir réussi la question précédente.

**Par exemple:**

| Test | Résultat |
|---|---|
| `// Méthodes de la classe de base ok`<br>`EtudiantCyclesSup grad(1337, "yolo", nullptr);`<br>`const Etudiant& etud = grad;`<br>`cout << etud.getMatricule() << endl`<br>`  << etud.getProgramme() << endl;` | `1337`<br>`yolo` |
| `// Implémentation de get/set superviseur ok`<br>`Professeur prof1, prof2;`<br>`EtudiantCyclesSup grad(1337, "yolo", &prof1);`<br>`cout << (grad.getSuperviseur() == &prof1) << endl;`<br>`grad.setSuperviseur(&prof2);`<br>`cout << (grad.getSuperviseur() == &prof2) << endl;` | `true`<br>`true` |
| `// Implémentation de changerProgramme ok`<br>`Professeur prof1;`<br>`EtudiantCyclesSup grad(1337, "googoo", &prof1);`<br>`Etudiant& etud = grad;`<br>`cout << etud.getProgramme() << endl;`<br>`etud.changerProgramme("gaga");`<br>`cout << etud.getProgramme() << endl;` | `googoo`<br>`pls halp`<br>`gaga` |

**Réponse :** (régime de pénalités : 0 %)

Réinitialiser la réponse

```
class EtudiantCyclesSup : public Etudiant {
    // Vous pouvez mettre les définitions de méthodes direct

    // TODO: Constructeur
    // TODO: getSuperviseur
    // TODO: setSuperviseur
    virtual void changerProgramme(const string nouveauProgra
private:
    Professeur* superviseur_;
};

```

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✖ | `// La classe hérite de Etudiant et n'est pas abstraite`<br>`cout << is_base_of_v<Etudiant, EtudiantCyclesSup> << endl`<br>`<<`<br>`is_abstract_v<EtudiantCyclesSup>`<br>`<< endl;` | true<br>false | `tests.cpp:13:360: error: non-static data member 'has_virtual_changerProgramme_v' declared 'constexpr'`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:360: error: 'has_virtual_changerProgramme_v' declared as an 'inline' field`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:365: error: data member 'has_virtual_changerProgramme_v' cannot be a member template`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:25:1: error: expected '}' at end of input`<br>`__tester__.cpp:33:18: error: 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)' marked 'override', but does not override`<br>`virtual void changerProgramme(const string nouveauProgramme)override{`<br>`^~~~~~~~~~~~~~~~`<br>`__tester__.cpp: In member function 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)':`<br>`__tester__.cpp:34:1: error: expected primary-expression before 'private'`<br>`private:`<br>`^~~~~~~`<br>`tests.cpp: In member function 'int EtudiantCyclesSup::main()':`<br>`tests.cpp:25:1: error: no return statement in function returning non-void [-Werror=return-type]`<br>`tests.cpp: At global scope:`<br>`tests.cpp:25:1: error: expected unqualified-id at end of input`<br>`cc1plus: some warnings being treated as errors` | ✖ |

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✖ | `// La classe a la bonne taille (la bonne quantité de données) cout << (sizeof(EtudiantCyclesSup) == tailleEtudiantCyclesSupAttendue) << endl;` | true | `tests.cpp:13:360: error: non-static data member 'has_virtual_changerProgramme_v' declared 'constexpr'`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:360: error: 'has_virtual_changerProgramme_v' declared as an 'inline' field`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:365: error: data member 'has_virtual_changerProgramme_v' cannot be a member template`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:24:1: error: expected '}' at end of input`<br>`__tester__.cpp:33:18: error: 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)' marked 'override', but does not override`<br>`virtual void changerProgramme(const string nouveauProgramme)override{`<br>`^~~~~~~~~~~~~~~~`<br>`__tester__.cpp: In member function 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)':`<br>`__tester__.cpp:34:1: error: expected primary-expression before 'private'`<br>`private:`<br>`^~~~~~~`<br>`tests.cpp: In member function 'int EtudiantCyclesSup::main()':`<br>`tests.cpp:24:1: error: no return statement in function returning non-void [-Werror=return-type]`<br>`tests.cpp: At global scope:`<br>`tests.cpp:24:1: error: expected unqualified-id at end of input`<br>`cc1plus: some warnings being treated as errors` | ✖ |

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✖ | `// Méthodes de la classe de base ok`<br>`EtudiantCyclesSup grad(1337, "yolo", nullptr);`<br>`const Etudiant& etud = grad;`<br>`cout << etud.getMatricule() << endl`<br>`<< etud.getProgramme() << endl;` | 1337<br>yolo | `tests.cpp:13:360: error: non-static data member 'has_virtual_changerProgramme_v' declared 'constexpr'`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:360: error: 'has_virtual_changerProgramme_v' declared as an 'inline' field`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:365: error: data member 'has_virtual_changerProgramme_v' cannot be a member template`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:27:1: error: expected '}' at end of input`<br>`__tester__.cpp:33:18: error: 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)' marked 'override', but does not override`<br>`virtual void changerProgramme(const string nouveauProgramme)override{`<br>`^~~~~~~~~~~~~~~~`<br>`__tester__.cpp: In member function 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)':`<br>`__tester__.cpp:34:1: error: expected primary-expression before 'private'`<br>`private:`<br>`^~~~~~~`<br>`tests.cpp: In member function 'int EtudiantCyclesSup::main()':`<br>`tests.cpp:23:45: error: no matching function for call to 'EtudiantCyclesSup::EtudiantCyclesSup(int, const char [5], std::nullptr_t)'`<br>`__tester__.cpp:27:7: note: candidate: EtudiantCyclesSup::EtudiantCyclesSup(const EtudiantCyclesSup&)`<br>`class EtudiantCyclesSup : public Etudiant {`<br>`^~~~~~~~~~~~~~~~~`<br>`__tester__.cpp:27:7: note: candidate expects 1 argument, 3 provided` | ✖ |

| Test | Résultat attendu | Résultat obtenu |
| --- | --- | --- |

```
__tester__.cpp:27:7: note: candidate:
EtudiantCyclesSup::EtudiantCyclesSup(EtudiantCyclesSup&&)
__tester__.cpp:27:7: note:   candidate expects 1 argument,
3 provided
tests.cpp:23:19: error: cannot declare variable 'grad' to
be of abstract type 'EtudiantCyclesSup'
__tester__.cpp:27:7: note:   because the following virtual
functions are pure within 'EtudiantCyclesSup':
 class EtudiantCyclesSup : public Etudiant {
       ^~~~~~~~~~~~~~~~~
__tester__.cpp:16:18: note: \tvirtual void
Etudiant::changerProgramme(const string&)
     virtual void changerProgramme(const string& programme)
= 0;
                  ^~~~~~~~~~~~~~~
tests.cpp:27:1: error: no return statement in function
returning non-void [-Werror=return-type]
tests.cpp: At global scope:
tests.cpp:27:1: error: expected unqualified-id at end of
input
cc1plus: some warnings being treated as errors
```

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✖ | `// Implémentation de get/set superviseur ok`<br>`Professeur prof1, prof2;`<br>`EtudiantCyclesSup grad(1337, "yolo", &prof1);`<br>`cout << (grad.getSuperviseur() == &prof1) << endl;`<br>`grad.setSuperviseur(&prof2);`<br>`cout << (grad.getSuperviseur() == &prof2) << endl;` | `true`<br>`true` | `tests.cpp:13:360: error: non-static data member 'has_virtual_changerProgramme_v' declared 'constexpr'`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:360: error: 'has_virtual_changerProgramme_v' declared as an 'inline' field`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:365: error: data member 'has_virtual_changerProgramme_v' cannot be a member template`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:28:1: error: expected '}' at end of input`<br>`__tester__.cpp:33:18: error: 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)' marked 'override', but does not override`<br>`virtual void changerProgramme(const string nouveauProgramme)override{`<br>`^~~~~~~~~~~~~~~~`<br>`__tester__.cpp: In member function 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)':`<br>`__tester__.cpp:34:1: error: expected primary-expression before 'private'`<br>`private:`<br>`^~~~~~~`<br>`tests.cpp: In member function 'int EtudiantCyclesSup::main()':`<br>`tests.cpp:24:44: error: no matching function for call to 'EtudiantCyclesSup::EtudiantCyclesSup(int, const char [5], Professeur*)'`<br>`__tester__.cpp:27:7: note: candidate: EtudiantCyclesSup::EtudiantCyclesSup(const EtudiantCyclesSup&)`<br>`class EtudiantCyclesSup : public Etudiant {`<br>`^~~~~~~~~~~~~~~~~`<br>`__tester__.cpp:27:7: note: candidate expects 1 argument, 3 provided` | ✖ |

| Test | Résultat attendu | Résultat obtenu |
| --- | --- | --- |

```
__tester__.cpp:27:7: note: candidate:
EtudiantCyclesSup::EtudiantCyclesSup(EtudiantCyclesSup&&)
__tester__.cpp:27:7: note:   candidate expects 1 argument,
3 provided
tests.cpp:24:19: error: cannot declare variable 'grad' to
be of abstract type 'EtudiantCyclesSup'
__tester__.cpp:27:7: note:   because the following virtual
functions are pure within 'EtudiantCyclesSup':
 class EtudiantCyclesSup : public Etudiant {
       ^~~~~~~~~~~~~~~~~
__tester__.cpp:16:18: note: \tvirtual void
Etudiant::changerProgramme(const string&)
     virtual void changerProgramme(const string& programme)
= 0;
                  ^~~~~~~~~~~~~~~
tests.cpp:25:15: error: 'class EtudiantCyclesSup' has no
member named 'getSuperviseur'
tests.cpp:26:6: error: 'class EtudiantCyclesSup' has no
member named 'setSuperviseur'
tests.cpp:27:15: error: 'class EtudiantCyclesSup' has no
member named 'getSuperviseur'
tests.cpp:28:1: error: no return statement in function
returning non-void [-Werror=return-type]
tests.cpp: At global scope:
tests.cpp:28:1: error: expected unqualified-id at end of
input
cc1plus: some warnings being treated as errors
```

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✖ | `// Implémentation de changerProgramme ok`<br>`Professeur prof1;`<br>`EtudiantCyclesSup grad(1337, "googoo", &prof1);`<br>`Etudiant& etud = grad;`<br>`cout << etud.getProgramme() << endl;`<br>`etud.changerProgramme("gaga");`<br>`cout << etud.getProgramme() << endl;` | googoo<br>pls<br>halp<br>gaga | `tests.cpp:13:360: error: non-static data member 'has_virtual_changerProgramme_v' declared 'constexpr'`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:360: error: 'has_virtual_changerProgramme_v' declared as an 'inline' field`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:365: error: data member 'has_virtual_changerProgramme_v' cannot be a member template`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:29:1: error: expected '}' at end of input`<br>`__tester__.cpp:33:18: error: 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)' marked 'override', but does not override`<br>`virtual void changerProgramme(const string nouveauProgramme)override{`<br>`^~~~~~~~~~~~~~~~`<br>`__tester__.cpp: In member function 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)':`<br>`__tester__.cpp:34:1: error: expected primary-expression before 'private'`<br>`private:`<br>`^~~~~~~`<br>`tests.cpp: In member function 'int EtudiantCyclesSup::main()':`<br>`tests.cpp:24:46: error: no matching function for call to 'EtudiantCyclesSup::EtudiantCyclesSup(int, const char [7], Professeur*)'`<br>`__tester__.cpp:27:7: note: candidate: EtudiantCyclesSup::EtudiantCyclesSup(const EtudiantCyclesSup&)`<br>`class EtudiantCyclesSup : public Etudiant {`<br>`^~~~~~~~~~~~~~~~~`<br>`__tester__.cpp:27:7: note: candidate expects 1 argument, 3 provided` | ✖ |

| Test | Résultat attendu | Résultat obtenu |
| --- | --- | --- |

```
__tester__.cpp:27:7: note: candidate:
EtudiantCyclesSup::EtudiantCyclesSup(EtudiantCyclesSup&&)
__tester__.cpp:27:7: note:   candidate expects 1 argument,
3 provided
tests.cpp:24:19: error: cannot declare variable 'grad' to
be of abstract type 'EtudiantCyclesSup'
__tester__.cpp:27:7: note:   because the following virtual
functions are pure within 'EtudiantCyclesSup':
 class EtudiantCyclesSup : public Etudiant {
       ^~~~~~~~~~~~~~~~~
__tester__.cpp:16:18: note: \tvirtual void
Etudiant::changerProgramme(const string&)
     virtual void changerProgramme(const string& programme)
= 0;
                  ^~~~~~~~~~~~~~~~
tests.cpp:29:1: error: no return statement in function
returning non-void [-Werror=return-type]
tests.cpp: At global scope:
tests.cpp:29:1: error: expected unqualified-id at end of
input
cc1plus: some warnings being treated as errors
```

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✖ | `// Suit les bonnes pratiques de redéfinition de méthodes` | `true` | `tests.cpp:13:360: error: non-static data member 'has_virtual_changerProgramme_v' declared 'constexpr'`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:360: error: 'has_virtual_changerProgramme_v' declared as an 'inline' field`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:13:365: error: data member 'has_virtual_changerProgramme_v' cannot be a member template`<br>`tests.cpp:15:1: note: in expansion of macro 'DEF_HAS_VIRTUAL'`<br>`tests.cpp:23:1: error: expected '}' at end of input`<br>`__tester__.cpp:33:18: error: 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)' marked 'override', but does not override`<br>`virtual void changerProgramme(const string nouveauProgramme)override{`<br>`^~~~~~~~~~~~~~~~`<br>`__tester__.cpp: In member function 'virtual void EtudiantCyclesSup::changerProgramme(std::__cxx11::string)':`<br>`__tester__.cpp:34:1: error: expected primary-expression before 'private'`<br>`private:`<br>`^~~~~~~`<br>`tests.cpp: In member function 'int EtudiantCyclesSup::main()':`<br>`tests.cpp:23:1: error: no return statement in function returning non-void [-Werror=return-type]`<br>`tests.cpp: At global scope:`<br>`tests.cpp:23:1: error: expected unqualified-id at end of input`<br>`cc1plus: some warnings being treated as errors` | ✖ |

## Solution de l'auteur de la question (Cpp):

```cpp
class EtudiantCyclesSup : public Etudiant {
public:
    EtudiantCyclesSup(int matricule, const string& progra
    : Etudiant(matricule, programme),
      superviseur_(superviseur) { }

    Professeur* getSuperviseur() const {
        return superviseur_;
    }
    void setSuperviseur(Professeur* superviseur) {
        superviseur_ = superviseur;
    }
    void changerProgramme(const string& programme) overri
        cout << "pls halp" << endl;
        programme_ = programme;
    }

private:
    Professeur* superviseur_;
};

```

Incorrect

Note pour cet envoi : 0,00/2,50.

Question **4**

Partiellement correct

Note de 0,90 sur 1,50

Nous avons la hiérarchie des classes suivantes qui représente des animaux. Un mulet est la progéniture hybride d'un âne et d'une jument.

 Animal en haut Cheval hérite de Animal Poney hérite de Cheval Âne hérite de Animal Mulet hérite de Cheval et de Âne

On voit facilement un problème d'héritage en diamant pour `Mulet`. Dans le code ci-dessous, remplissez les boîtes avec le type d'héritage approprié pour que les classes soient compilables et utilisables. Si vous avez à le faire, mettez le moins d'héritage virtuel possible.

```
class Animal {
public:
    Animal(int poids)
    : poids_(poids) {
        printf("Animal(%i)" "\n", poids);
    }

    virtual ~Animal() {
        cout << "~Animal()" << "\n";
    }

    int getPoids() const { return poids_; }
    void setPoids(int poids) { poids_ = poids; }

    virtual void vivre() = 0;

private:
    int poids_;
};

class Cheval : public virtual ✔ Animal {
public:
    Cheval(int poids)
    : Animal(poids) {
        printf("Cheval(%i)" "\n", poids);
    }

    ~Cheval() override {
        cout << "~Cheval()" << "\n";
    }

    void vivre() override {
        cout << "Sometimes life's a bitch and then you keep living." << "\n";
    }
};

class Poney : public ✔ Cheval {
public:
    Poney(int poids)
    : Animal(poids),
      Cheval(poids) {
        printf("Poney(%i)" "\n", poids);
    }
```

```
44.
45.     ~Poney() override {
46.         cout << "~Poney()" << "\n";
47.     }
48. };
49.
50. class Ane : public virtual ✔ Animal {
51. public:
52.     Ane(int poids)
53.     : Animal(poids) {
54.         printf("Ane(%i)" "\n", poids);
55.     }
56.
57.     ~Ane() override {
58.         cout << "~Ane()" << "\n";
59.     }
60.
61.     void vivre() override {
62.         cout << "Alright, nobody move! I've got a dragon here, and I'm not afraid to use it!" << "\n";
63.     }
64. };
65.
66. class Mulet : public virtual ✖ Cheval, public virtual ✖ Ane {
67. public:
68.     Mulet(int poids)
69.     : Animal(poids),
70.       Cheval(poids),
71.       Ane(poids) {
72.         printf("Mulet(%i)" "\n", poids);
73.     }
74.
75.     ~Mulet() override {
76.         cout << "~Mulet()" << "\n";
77.     }
78.
79.     void vivre() override {
80.         Ane::vivre();
81.     }
82. };
```

Description

Indiquez l'ordre de complexité de temps des algorithmes ci-dessous.  On considère l'ordre moyen et/ou amorti.  S'il y a plusieurs possibilités, indiquez l'ordre qui est le plus petit possible.

La fonction *f* a un temps en *O*(1) et *range* n'ajoute pas de complexité par rapport à une boucle faite à la main.

Question **5**

Incorrect

Note de 0,00 sur 1,25

```
map<int,int> v;
for (int i : range(n))
    v[f(i)] = i;
```

Est *O*( n ✖ )

Votre réponse est incorrecte.

La réponse correcte est :

```
map<int,int> v;
for (int i : range(n))
    v[f(i)] = i;
```

Est *O*([n · log👎 ])

Question **6**

Incorrect

Note de 0,00 sur 1,25

```
string v = "x";
for (int i : range(n))
    v = "hello " + v;
```

Est $O($ n ✖ $)$

Votre réponse est incorrecte.

La réponse correcte est :

```
string v = "x";
for (int i : range(n))
    v = "hello " + v;
```

Est $O([n^2])$

Question **7**

Correct

Note de 2,00 sur 2,00

Remplissez la fonction suivante qui trouve les multiples d'un entier donné dans un vecteur d'entiers donné. Vous devez faire cette recherche en une seule ligne de code en utilisant les algorithmes et adaptateurs de la librairie standard. Vous ne pouvez pas faire de boucle à la main. N'oubliez pas que vous avez accès à `<algorithm>` et à `<iterator>`, incluant les adaptateurs d'itérateurs.

**Par exemple:**

| Test | Résultat |
| --- | --- |
| `vector<int> foo = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};`<br>`printVector(trouverMultiples(foo, 2));`<br>`cout << endl;`<br>`printVector(trouverMultiples(foo, 3));` | `[2 4 6 8 10 12]`<br>`[3 6 9 12]` |

**Réponse :** (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1 ▾ vector<int> trouverMultiples(const vector<int>& valeurs, int
2       std::vector<int> resultat;
3       std::copy_if(valeurs.begin(), valeurs.end(), std::back_in
4       return resultat;
5   }
```

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✔ | `vector<int> foo = {1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13};`<br>`printVector(trouverMultiples(foo, 2));`<br>`cout << endl;`<br>`printVector(trouverMultiples(foo, 3));` | `[2 4 6 8 10 12]`<br>`[3 6 9 12]` | `[2 4 6 8 10 12]`<br>`[3 6 9 12]` | ✔ |

Tous les tests ont été réussis ! ✔

## Solution de l'auteur de la question (Cpp):

```cpp
vector<int> trouverMultiples(const vector<int>& valeurs, int
    vector<int> resultat;
    copy_if(
        valeurs.begin(),
        valeurs.end(),
        back_inserter(resultat),
        [&] (int val) { return val % base == 0; }
    );
    // Alternative :
    //for_each(
    //    valeurs.begin(),
    //    valeurs.end(),
    //    [&] (int val) {
    //        if (val % base == 0)
    //            resultat.push_back(val);
    //    }
    //);
    return resultat;
}
```

Correct

Note pour cet envoi : 2,00/2,00.

Question **8**

Incorrect

Note de 0,00 sur 2,00

Remplissez la fonction suivante qui supprime, dans un vecteur de réels donné, tous les nombres ne faisant pas partie de la plage donnée. Vous devez faire toute cette opération en une ou deux lignes de code et en utilisant les algorithmes de la librairie standard. Vous ne pouvez pas faire de boucle à la main. N'oubliez pas que vous avez accès à la librairie `<algorithm>`.

**Par exemple:**

| Test | Résultat |
| --- | --- |
| `vector<double> foo = {0.1337, 0.69, 0.42, 1.5, 9000, 420.69};`<br>`effacerSortantPlage(foo, 0, 1000);`<br>`printVector(foo); cout << endl;`<br>`effacerSortantPlage(foo, 0.2, 2);`<br>`printVector(foo); cout << endl;` | `[0.1337 0.69 0.42 1.5 420.69]`<br>`[0.69 0.42 1.5]` |

**Réponse :** (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1  void effacerSortantPlage(vector<double>& valeurs, double min,
2  {
3      // Enlever de `valeurs` tous les éléments qui ne sont pas
4  }
```

| | Test | Résultat attendu | Résultat obtenu | |
|---|---|---|---|---|
| ✖ | `vector<double> foo = {0.1337, 0.69, 0.42, 1.5, 9000, 420.69};`<br>`effacerSortantPlage(foo, 0, 1000);`<br>`printVector(foo); cout << endl;`<br>`effacerSortantPlage(foo, 0.2, 2);`<br>`printVector(foo); cout << endl;` | `[0.1337 0.69 0.42 1.5 420.69]`<br>`[0.69 0.42 1.5]` | `[0.1337 0.69 0.42 1.5 9000 420.69]`<br>`[0.1337 0.69 0.42 1.5 9000 420.69]` | ✖ |

Montrer les différences

## Solution de l'auteur de la question (Cpp):

```cpp
void effacerSortantPlage(vector<double>& valeurs, double min,
    auto it = remove_if(
        valeurs.begin(),
        valeurs.end(),
        [&] (double val) { return val < min or val > max; }
    );
    valeurs.erase(it, valeurs.end());
}
```

Incorrect

Note pour cet envoi : 0,00/2,00.

Question **9**

Non répondue

Note de 0,00 sur 2,25

Complétez le code fournis pour implémenter un simple gestionnaire de commande générique qui associe un nom (une string) à une action (une fonction). Les fonctions qui sont enregistrées doivent prendre un paramètre du type spécifié au gestionnaire et ne rien retourner (void).

Vous devez utiliser la librairie standard pour contenir les fonctions et les associations. Vous devez supporter **tout type d'objet fonctionnel** comme action de commande (fonction, lambda, foncteur). Aussi, la recherche d'une commande par son nom doit avoir une **complexité constante** et **l'ordre à l'intérieur du conteneur n'est pas important**.

**Par exemple:**

| Test | Résultat |
|---|---|
| `struct Foncteur {`<br>`  Foncteur(int m) : m_(m) { }`<br>`  void operator()(int x) {`<br>`  cout << (x + m_) << endl;`<br>`  }`<br>`  int m_;`<br>`};`<br><br>`GestionnaireCommandes<int> foo;`<br>`Foncteur f10(10);`<br>`Foncteur f100(100);`<br>`foo.ajouterCommande("plus10", f10);`<br>`foo.ajouterCommande("plus100", f100);`<br>`auto plus10 = foo.trouverCommande("plus10").value();`<br>`auto plus100 = foo.trouverCommande("plus100").value();`<br>`plus10(420);`<br>`plus100(420);`<br>`plus10(69);`<br>`plus100(69);` | 430<br>520<br>79<br>169 |

**Réponse :** (régime de pénalités : 0 %)

Réinitialiser la réponse

```
1  template <typename TypeParametre>
2  class GestionnaireCommandes {
3      friend int main(); // NE PAS TOUCHER À CETTE LIGNE! C'es
4
5  public:
```

```
    using TypeFonction = /* TODO: Le type de fonction à cont

    void ajouterCommande(const string& nomCommande, TypeFonc
        // TODO: Ajouter l'action à commandes_ avec le nom d
    }

    optional<TypeFonction> trouverCommande(const string& nom
        // TODO: Si la commande est présente dans commandes_
        //       sinon retourner un optional vide.
    }

private:
    /* TODO: Le type de conteneur associant le nom (string)
};
```

## Solution de l'auteur de la question (Cpp):

```
template <typename TypeParametre>
class GestionnaireCommandes {
    friend int main(); // NE PAS TOUCHER À CETTE LIGNE! C

public:
    using TypeFonction = function<void (TypeParametre)>;

    void ajouterCommande(const string& nomCommande, TypeF
        // TODO: Ajouter l'action à commandes_ avec le no
        commandes_[nomCommande] = action;
    }

    optional<TypeFonction> trouverCommande(const string&
        // TODO: Si la commande est présente dans command
        //       sinon retourner un optional vide.
        if (commandes_.count(nomCommande) != 0)
            return commandes_.at(nomCommande);
        return {};
    }
```

Question **10**

Non répondue

Noté sur 1,25

Dans le modèle MVC, le contrôleur. . . (points retirés par mauvaise réponse)

Veuillez choisir au moins une réponse.

- a. ... modifie le modèle suite à certains événements
- b. ... vérifie la validité des actions de l'utilisateur
- c. ... formate les données
- d. ... modifie la vue suite à certains événements
- e. ... maintient l'état des données
- f. ... gère tous les événements de toutes les sources

Votre réponse est incorrecte.

Les réponses correctes sont : ... modifie le modèle suite à certains événements, ... modifie la vue suite à certains événements, ... vérifie la validité des actions de l'utilisateur

Description

Soit le code suivant qui calcule la variance et l'asymétrie d'une loi binomiale en fonction d'un *n* (nombre d'expériences) et d'un *p* (probabilité de succès) donné. Notez qu'un *p* d'exactement 0 ou 1 donne une variance de 0. On définit aussi les types d'exceptions qui peuvent être lancées par ces fonctions.

```
1. struct ErreurN : public domain_error {
2.     using domain_error::domain_error;
3. };
4.
5. struct ErreurP : public domain_error {
6.     using domain_error::domain_error;
7. };
8.
9. struct ErreurDivZero : public domain_error {
10.    using domain_error::domain_error;
11. };
12.
13. double calculerVariance(int n, double p) {
14.     if (n <= 0)
15.         throw ErreurN("n="s + to_string(n) + " n'est pas un nombre naturel"s);
16.     if (p < 0 or p > 1)
17.         throw ErreurP("p="s + to_string(p) + " n'est pas dans l'intervalle [0, 1]");
18.     return n * p * (1 - p);
19. }
20.
21. double calculerAsymetrie(int n, double p) {
22.     double variance = calculerVariance(n, p);
23.     if (variance == 0)
24.         throw ErreurDivZero("Variance ne peut pas être nulle.");
25.     double q = 1 - p;
26.     return (q - p) / sqrt(variance);
27. }
```

Question **11**

Non répondue

Noté sur 1,50

Décrivez le comportement du programme suivant..

```
1.  int main() {
2.      try {
3.          int n = 42;
4.          double p = 0.0;
5.          double asym = calculerAsymetrie(n, p);
6.          cout << "n=" << n << " p=" << p << " var=" << asym << endl;
7.      } catch (ErreurN& e) {
8.          cout << "Problème avec nombre d'expériences : " << e.what() << endl;
9.      } catch (ErreurP& e) {
10.         cout << "Problème avec probabilité : " << e.what() << endl;
11.     } catch (logic_error& e) {
12.         cout << "Erreur logique : " << e.what() << endl;
13.     }
14. }
```

a. Le calcul de l'asymétrie échoue à cause de la valeur de la variance qui est zéro, une exception est lancée dans calculerAsymétrie() et est attrapée par le troisième catch. Le programme se termine normalement (atteint la fin du main).

b. Le calcul de l'asymétrie échoue à cause de la valeur de la variance qui est zéro, une exception est lancée dans calculerAsymétrie() et est attrapée par le troisième catch après avoir affiché le résultat erroné du calcul (exécuté la ligne 6). Le programme se termine normalement (atteint la fin du main).

c. Le calcul de l'asymétrie échoue à cause de la valeur de *p*, une exception est lancée dans calculerVariance() et est attrapée par le premier catch. Une deuxième exception est ensuite lancée dans calculerAsymetrie() à cause de la variance qui est nulle (*p*=0). Cette exception n'est pas attrapée et le programme se termine abruptement.

Votre réponse est incorrecte.

La réponse correcte est : Le calcul de l'asymétrie échoue à cause de la valeur de la variance qui est zéro, une exception est lancée dans calculerAsymétrie() et est attrapée par le troisième catch. Le programme se termine normalement (atteint la fin du main).